NEW VERSION:
SkePU v1.1 May 2014

SkePU

Auto-tunable Multi-Backend Skeleton Programming Library for Multi-GPU Systems

Usman Dastgeer **Johan Enmyren** **Christoph Kessler**

Linköping University, Sweden

Multi-GPU Systems

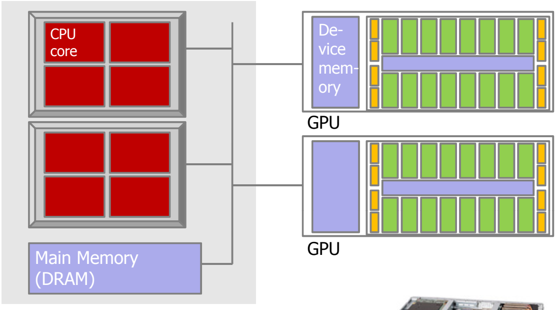


Image source: Supermicro®, www.supermicro.com

Challenges

- Portability
- Programmability
- Performance portability

OpenCL™ ? Portable but low-level code…

Solution: Skeleton Programming

Skeletons are pre-defined, reusable, parameterizable generic components with well defined semantics, for which efficient parallel or accelerator-specific implementations may be available.

- Higher-order functions
- for frequent algorithmic (control and data flow) patterns e.g. map, reduce, scan, farm, pipe, DC, …
- parameterized in sequential user code, programmer interface is sequential
- encapsulate all parallelism and platform-specific implementation details
- well suited for internal autotuning
- enforce well structured code

SkePU Example: Dot product

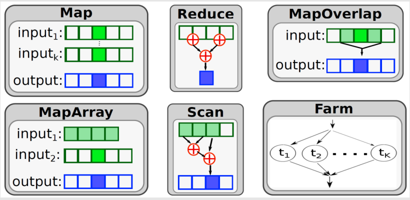
```
#include <iostream>

#include "skepu/vector.h"
#include "skepu/mapreduce.h"

BINARY_FUNC(plus, double, a, b,
    return a+b;
)

BINARY_FUNC(mult, double, a, b,
    return a*b;
)

….
```

Platform-specific code for user-defined functions generated from such macros

```
….

int main()
{
   skepu::MapReduce<mult, plus>
        dotProduct(new mult, new plus);

   skepu::Vector<double> v0(…);
   skepu::Vector<double> v1(…);

   double r = dotProduct( v0, v1 );

   std::cout << "Result: " << r << "\n";

   return 0;
}
```

Create a skeleton instance

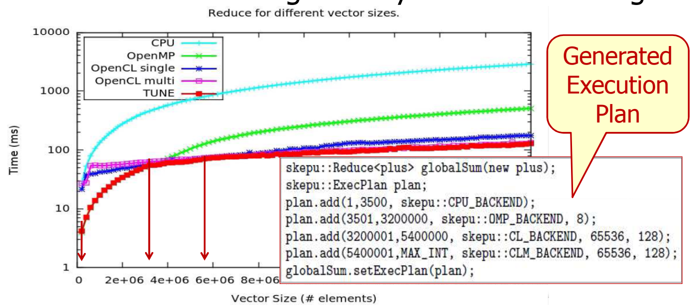
SkePU



- C++ template library
- 6 data-parallel skeletons
  - map, reduce, scan, mapreduce, maparray, mapoverlap (stencil)
- 1 task-parallel skeleton: farm
  - *All* skeleton calls execute asynchronously, synchronized automatically by operand data flow
- Smart containers for operand data: Vector, Matrix, …
  - Software caching of sub-arrays in device memories, sequentially consistent
  - Optimizing PCIe communication and memory management at runtime
- Generation of platform-specific user functions
- Back-ends for C, OpenMP, OpenCL, CUDA, StarPU, (MPI)
- Multi-GPU support
- Overhead below 10%
- Tunable

Offline-tunable selection

- Expected best back-end + tunable parameters: #threads, #thread blocks, tiling factor, …
- Execution plan for each skeleton, generated from off-line training data by machine learning



Generated Execution Plan

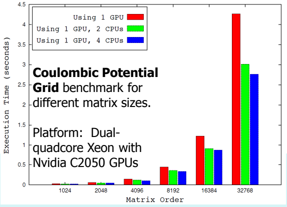
```
skepu::Reduce<plus> globalSum(new plus);
skepu::ExecPlan plan;
plan.add(1,3500, skepu::CPU_BACKEND);
plan.add(3501,3200000, skepu::OMP_BACKEND, 8);
plan.add(3200001,5400000, skepu::CL_BACKEND, 65536, 128);
plan.add(5400001,MAX_INT, skepu::CLM_BACKEND, 65536, 128);
globalSum.setExecPlan(plan);
```

On-line tunable selection

- Using the history-guided selection in StarPU

Hybrid execution

- CPUs + GPUs in parallel



Coulombic Potential Grid benchmark for different matrix sizes.

Platform: Dual-quadcore Xeon with Nvidia C2050 GPUs

Selected publications

- J. Enmyren, C. Kessler: **SkePU: A Multi-Backend Skeleton Programming Library for Multi-GPU Systems.** Proc. 4th Int. Worksh. on High-Level Parallel Progr. and Applications (HLPP-2010), Baltimore, USA, Sep. 2010. ACM.
- U. Dastgeer, J. Enmyren, C. Kessler: **Auto-tuning SkePU: A Multi-Backend Skeleton Programming Framework for Multi-GPU Systems.** Proc. IWMSE-2011, Hawaii, USA, May 2011, ACM.
- U. Dastgeer, C. Kessler, S. Thibault. **Flexible runtime support for efficient skeleton programming on hybrid systems**. ParCo'11: Int. Conf. on Parallel Computing. Ghent, Belgium, 2011.
- U. Dastgeer, C. Kessler: **Smart Containers and Skeleton Programming for GPU-based systems**. HLPP'14, Amsterdam, NL, July 2014.

Open Source:

www.ida.liu.se/~chrke/skepu